Web-Interface Plan

<u>Overview of the Web Interface</u>

We are developing a web-based interface to allow users to explore our annotated movie review corpus interactively. The main objectives of the interface are:

- Search and retrieve movie reviews from our corpus based on user input.
- Allow users to view annotated reviews, filtering by rhetorical strategy (Logos, Pathos, Ethos).
- Enable interactive exploration with a clean and user-friendly interface.

To achieve this, we will build:

- A FastAPI backend to handle data retrieval and processing.
- A frontend using HTML, JavaScript (Vue.js or React), and CSS for user interaction.
- A lightweight search system (Whoosh) or Elasticsearch to support search functionality.

<u>Core Features and Functionality</u>

The interface will support the following key features:

1. Search Functionality

Users will be able to search the movie review corpus using the following options:

- Keyword Search: Users can enter a word or phrase to retrieve relevant reviews.
- Filter by Annotation: Users can filter results by rhetorical strategy (Logos, Pathos, Ethos).
- Sort Results: (Optional) Users can sort reviews by review length or sentiment (if applicable).

Implementation:

- The backend will index the corpus using Whoosh (lightweight) or Elasticsearch (if scalability is needed).
- Search queries will be handled via a FastAPI endpoint (/search).
- The frontend will include a search bar and filter options to refine results.

2. Displaying Annotated Reviews

Since not all reviews are annotated, users will have an option to:

- Toggle a checkbox (“Annotated Data Only”) to view only annotated reviews.
- View the annotation label (Logos, Pathos, Ethos) next to each result.

Implementation:

- The annotation data is stored in a CSV or JSON format, linked to each review by review_id.

- A FastAPI endpoint (/annotations) will return annotation labels for requested reviews.

Example JSON response:

{"review_id": "001", "review_text": "This film was breathtaking!", "annotation_label": "Logos"}

3. User Interface (Frontend)

The interface will be minimalist, responsive, and user-friendly:

- A search bar at the top for entering keywords.
- A filter panel to refine search results (by annotation type).
- A results display area, showing reviews and their labels (if annotated).

Implementation:

- We will use Vue.js or React to create a dynamic and interactive front-end.
- The frontend will communicate with the FastAPI backend via AJAX requests.

<u>Technical Implementation Plan</u>

Backend (FastAPI)

- Handles search queries and filters results using Whoosh/Elasticsearch.
- Provides annotation data via API endpoints (/search and /annotations).
- Manages user requests for filtered search and annotation retrieval.

FastAPI Endpoints

Endpoint	**Function**
/search	Retrieves reviews based on user input
/annotations	Returns annotation data for a given review
/download	Allows users to download the annotated dataset

<u>Frontend (Vue.js or React)</u>

- Search bar & filter options to refine queries.
- AJAX requests to fetch search results dynamically.
- Results display with review text and annotation labels.

User Workflow

1. User enters a search term (e.g., “cinematography”).
2. The interface fetches matching reviews from the backend.
3. If “Annotated Data Only” is checked, only labeled reviews are shown.
4. Users can click on a review for additional details (e.g., full text, metadata).

Justification of Design Choices

Why FastAPI?

- High performance and lightweight for API-based corpus retrieval.
- Asynchronous capabilities for fast query responses.

Why Whoosh/Elasticsearch?

- Whoosh: Simple, easy-to-integrate full-text search.
- Elasticsearch: More scalable for larger datasets.

Why Vue.js/React?

- Fast, interactive, and responsive UI.
- Seamless integration with FastAPI for real-time updates.